NADA, KTH
Master's thesis
2002/2003

JAVA APPLICATIONS IN MOBILE DEVICES

- concerning business services at Telia Mobile AB

Author: Mattias Hellström

Supervisor: Henrik Eriksson

Abstract

I have investigated the new Java technology for mobile phones, J2ME (Java 2 Micro Edition), particularly with respect to the business services Telia Mobile AB offers on the Swedish market. By studies of J2ME, the GSM network, the GPRS network and Telia Mobile's business services in Sweden I have identified some areas where J2ME could ease the usage and reduce the time to activate features of services. I have also developed an application that can activate a "Mobile Reference" for a Dirigent subscriber. The application works in a production environment. I tried to make a similar application that would work with the equivalent feature for a Centrex subscriber. Unfortunately, due to technical limitations of J2ME and the limitations of the Java enabled phones on the market today this could not be done. I have also investigated the possibilities and limitations of J2ME.

Finally, I have come to some conclusions and recommendations for Telia Mobile on how Telia Mobile should tackle the technology in order to take advantage of it. In the short run I don't think the technology will affect the market a lot, the reason being that the technology is not perfect, that the knowledge about the technology within the business is limited not only among the customers, but also at big, leading telecom companies, and that most customers do not have Java enabled mobile phones. However, in the long run I think that this is a technology that the customers will demand, so Telia Mobile should start to develop strategies in order to take advantage of the technology. One aspect is the distribution and billing system of Java applications. I have some proposals that Telia Mobile should look into. As the lack of knowledge concerning J2ME is extensive among the customers, something should be done to change the situation. To do that I have some recommendations that hopefully could improve the knowledge.

Keywords: J2ME, MIDP, CLDC, Dirigent, Centrex, Mobile Switcher

JAVAAPPLIKATIONER I MOBILA TERMINALER

- med avseende på företagstjänster som Telia Mobile AB erbjuder

Sammanfattning

Jag har utrett den nya Javatekniken för mobiltelefoner, J2ME (Java 2 Micro Edition), främst med avseende på de företagstjänster som Telia Mobile AB erbjuder inom Sverige. Med hjälp av studier av J2ME, GSM nätet, GPRS nätet och Telia Mobiles företagstjänster i Sverige har jag identifierat några områden där J2ME skulle kunna förenkla användandet samt minska tiden det tar att aktivera delar av tjänster. Jag har även utvecklat en applikation som kan aktivera Dirigents ”Mobil Hänvisning” och som fungerar i skarp miljö. Jag försökte göra en liknande applikation med samma funktion för Centrexs ”Mobil Hänvisning”, men tekniska begränsningar i J2ME samt i de mobiltelefoner som finns på marknaden idag gjorde att den inte fungerade. Jag har även utrett vilka möjligheter samt begränsningar J2ME har.

Avslutningsvis har jag kommit fram till en del rekommendationer om hur Telia Mobile ska förhålla sig till tekniken för att utnyttja den på bästa sätt. På kort sikt tror jag inte att tekniken kommer att påverka marknaden nämnvärt. Det beror bland annat på att tekniken inte är fulländad och att kunskapen om tekniken är begränsad, inte bara bland kunder utan även hos ledande företag i branschen. Det beror också på att de flesta kunder inte har mobiltelefoner med Javastöd i. Jag tror dock att det är en teknik som de flesta kunder kommer att ta till sig på lite längre sikt. Därför bör Telia Mobile lägga upp strategier för att kunna utnyttja teknologin på bästa sätt. En aspekt är distributionen och betalningsprocessen av Javaapplikationer. Där har jag några förslag som Telia Mobile borde titta närmare på. Eftersom okunskapen rörande J2ME är omfattande bland kunderna måste någonting göras för att ändra på det. Jag har några förslag som förhoppningsvis kan förbättra situationen.

Nyckelord: J2ME, MIDP, CLDC, Dirigent, Centrex, Mobile Switcher

Acknowledgements

This master's thesis has been written at the departement of Service Design at Telia Mobile AB for Telia Mobile AB and NADA, KTH. The project has been produced in close cooperation with MSc student Daniel Hellstrand. My supervisors at Telia Mobile have been MSc Robert Tjernström and MSc Markus Marklund. My supervisor at KTH has been Doctor Henrik Eriksson. I would like to thank my supervisors at Telia Mobile and the whole Service Design department for being supportive and always helping, whenever I had questions. I learned a lot from them and I really appreciated that they always put away their work to explain different complex technical problems to me, when I came with a question. It was fun to go to work and I think the colleagues at the department are of great importance for that. I will always remember our laughter in the corridor and by the coffee table. I also would like to thank my enthusiastic supervisor at KTH, Lector Henrik Eriksson for good ideas, which helped me focus on the important issues. Finally, I would like to thank MSc student Daniel Hellstrand for interesting and creative discussions and a great support, particularly when it comes to the implementations. Without these people I would never had come this far.

SUNDSVALL, 2003-02-18

Mattias Hellström

Table of contents

1 BACKGROUND AND PURPOSE 1

1.1 BACKGROUND 1
1.2 PURPOSE 1
1.3 METHOD 1
1.4 OBJECTIVES 2
1.5 DELIMITATIONS 2

2 THEORY 3

2.1 JAVA 3
2.1.1 History 3
2.1.2 J2ME (Java 2 Micro Edition) 4
2.1.3 MIDP 1.0, 2.0, CDC and CLDC comparison 7
2.1.4 MIDP (Mobile Information Device Profile) 7
2.1.5 Restrictions within J2ME 8
2.1.6 Restrictions within MIDP 10
2.1.7 MIDP 1.0 10
2.1.8 MIDP 2.0 11
2.1.9 Java Servlet Technology 12
2.1.10 JSP (Java Server Pages Technology) 13
2.1.11 Optional Packages 13
2.1.12 JSR (Java Specification Request) 14
2.1.13 Smart cards 15
2.1.14 Java Card™ Technology 15
2.1.15 The Java Card API 17
2.1.16 Java phones in other countries, distribution of earnings 17
2.2 MOBILE NETWORK 18
2.2.1 GSM network with an IN node 18
2.2.2 GPRS (General Packet Radio Service) 20
2.2.3 UMTS (Universal Mobile Telecommunications System) 21
2.2.4 USSD (Unstructured Supplementary Service Data) 21
2.3 MOBILE BUSINESS SERVICES 22
2.3.1 Dirigent 22
2.3.2 Centrex 22
2.4 VIRUSES 23
2.4.1 Computer viruses 23
2.4.2 Mobile viruses 23

3 EMPIRY 24

3.1 MOBILE BUSINESS SERVICES 24
3.1.1 Dirigent 24
3.1.2 Centrex 27
3.2 J2ME 28
3.2.1 USSD 28
3.2.2 Servlets 28

4 ANALYSIS AND RECOMMENDATIONS 29

4.1 J2ME 29
4.1.1 Distribution and billing 29
4.1.2 Lack of knowledge 32
4.1.3 Japan 33
4.1.4 Development of J2ME and JSR 33
4.1.5 Games 33
4.1.6 What APIs are included? 33
4.1.7 Future services 34
4.1.8 USSD and development of new mobile services 34
4.1.9 Servlets and JSPs 34
4.1.10 Java Card Technology 34
4.1.11 Different applications for different mobile phones 35
4.1.12 WMA 35
4.1.13 MIDP 2.0 and MP3 35
4.1.14 Storage space online 35
4.2 MOBILE BUSINESS SERVICES 36
4.2.1 Dirigent 36
4.2.2 Centrex 36
4.3 SECURITY 36
4.3.1 Trojan horse 36
4.3.2 Data erased 36
4.3.3 Copying of applications 37
4.3.4 WAP Gateways 38

5 CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDIES 39

5.1 CONCLUSIONS 39
5.2 RECOMMENDATIONS FOR FURTHER STUDIES 39

6 REFERENCES 40

6.1 PRINTED BOOK 40
6.2 INTERNET 40
6.3 PERSONAL COMMUNICATION 42
6.4 OTHER 42

7 APPENDIX 43

1 Background and purpose

1.1 Background

Today there are quite a few mobile devices that support Java applications (http://www.microjava.com/devices, 2003-02-03) and the number will continue to rise. The mobile device manufacturers have indicated that a big percentage of the devices of the future will support the Java technology. Thanks to the integration of mobile telephony and the Internet it has become possible to download and install small applications in the mobile devices. This technology can be compared with the downloading of programs/applications from the Internet to a normal PC. The difference is only that the applications in the mobile devices are Java applications. These applications can be independent applications that are being run locally in the mobile device or alternatively, applications that are being run with a service server. For both variants one normally has to install them with the help of OTA-technology (Over The Air). With the new networks, GPRS (General Packet Radio Service) and UMTS (Universal Mobile Telecommunications System), these mobile devices become more interesting since the customers can be online constantly and only pay for the data that is being sent. The Java Technology can be used in existing services by downloading the adopted applications and installing them in the mobile device in order to facilitate the use.

1.2 Purpose

The purpose of the master's thesis is to understand the Java technology, how it can be used for mobile services and how it can communicate with the existing mobile network and its interface. A part of the purpose is to get an insight into the supply of services Telia Mobile AB is offering in the business segment. With this knowledge I will specify and develop an application that simplifies the use of the business services. Another part of the purpose is to come up with solutions and proposals that concern the possibilities and restrictions with Java technology in the mobile network.

1.3 Method

Since the Java Technology for mobile devices is a rather new technology there are not many books on the market covering neither the theory nor the different implications of it. Since a big part of my sources has been published on the Internet during the autumn 2002, i.e. at the same time as I have been writing this master's thesis, I have mostly used the Internet as information source.

I have searched for information by using Google and AltaVista, sending requests to persons at many different organizations, mostly other telecom companies, within Telia Mobile and to Sun Microsystem. The requests have been presented over the phone, by e-mail and whenever possible between four eyes. To show that the technology can be used today and that it may actually facilitate usage, I have specified and developed an application in close cooperation with MSc student Daniel Hellstrand. With support from the theory and empiry I have come to some conclusions about the Java Technology in mobile devices and some recommendations to Telia Mobile. The project has focused on business services since there was a need of this knowledge within Telia Mobile. However, I have also conclusions concerning Java Technology in mobile devices in general and Java Technology on the private market.

Telia Mobile has supported me with two mobile phones: a Siemens SL45i and a Nokia 7650 (see figure 1), which both support Java. Furthermore, I have been supported with some different mobile phone subscriptions, a computer and a room at the Service Design Department.

Figure 1. Nokia 7650
(http://www.nokia.se/mobiltelefoner/7650/#, 2003-02-03)

1.4 Objectives

The objectives of the master's project are:

- Understand how the Java technology can be used in mobile devices
- Present one or two services where Java applications could make the service easier to use and available for more customers
- Specify the functionality of the application and the effect on the mobile network
- Realize an application and test it on an appropriate business customer
- Investigate the possibilities and limitations with Java for mobile devices and specifically what it means for Telia Mobile

1.5 Delimitations

I have decided to delimit the master's project to the Java technology. I could have examined competing technologies, such as Symbian and other operating systems, and compared them to Java, but that lies outside of my project specification. I have also concentrated on the Java technology for mobile phones and not PDAs since most people have a mobile phone, but not a PDA.

2 Theory

The Java technology is first presented and then the mobile network GSM, the Intelligent Node, IN, and the GPRS network are described. Some business mobile services at Telia Mobile AB are then being introduced. Finally, the security aspect is covered with some former experiences.

2.1 Java

2.1.1 History

It all started 1991 when Patrick Naughton, Mike Sheridan and James Gosling started to create something that was supposed to be a programming tool at Sun. Creating a new language was never the goal. The secret "Green Team" consisting of 13 people had a conclusion saying that the convergence of digitally controlled consumer devices and computers would become a significant trend in the industry. With that in mind they created an interactive, handheld home entertainment device controller with an animated touch screen user interface. Since it ran on an entirely new processor-independent language it was able to control a wide range of entertainment platforms and appliances, while displaying animation. Green Team member James Gosling created the language specifically for the demo. He first called the language "Oak" after a tree outside his window. The whole project started involving potential customers in the cable television industry and the Green Team developed a new demo called MovieWood. It was interactive so that users could read and write information into the system. However, the companies in the cable television business did not want to lose that much control. Sun then realized that there was not a business in digital cable television.(http://java.sun.com/features/1998/05/birthday.html, 2002-09-25)

At a group meeting with Gosling and five other important persons from Sun they come up with the idea: "Why not the Internet?" Internet was becoming popular and it was exactly the type of network configuration that the former Green Team had envisioned for the cable TV industry. In fact, the Internet was being transformed into exactly the network that the Green Team had been trying to convince the cable companies they ought to be building. "It was just an incredible accident and it was patently obvious that the Internet and Java were a match made in heaven" according to Gosling. 1994 the team produced the HotJava™ browser, which for the first time brought to life animated, moving objects and dynamic executable content inside a Web browser. Later on, to be more precise early in 1995, these features were demonstrated to Internet and entertainment professionals at the Technology, Entertainment and Design Conference in Monterey. The audience was amazed by the new technology and the rumours about the new technology spread quickly. One month after the Conference in Monterey Goslings' group was ready to make the binary code available and free over the Internet. By March 1995 the number of binary copies outside Sun was only seven or eight, but just in a few months, the downloads began to surge into thousands. Gosling had set up a goal that stated that if they reached 10 000 downloads it would be "a total, blow-the-doors-off success". This goal was reached even faster then Gosling in his wildest dreams ever could imagine. With marketing and a front-page story in San Jose Mercury News, the downloading continued. (Byouns, 1998)

GARDENERS, THINK SPRING
FOR WOMEN ONLY
EDUCATION OVERHAUL
San Jose Mercury News
Why Sun thinks Hot Java will give you a lift
Blizzard
Mad rush for U.S. citizenship

Figure 2. The front-page of San Jose Mercury News, March 23, 1995. (http://java.sun.com/features/1998/05/birthday.html, 2002-09-25)

According to David Bank Sun gave away Java and HotJava free for non-commercial use in a fast-track attempt to make them the standard before Microsoft began shipping a similar product, codenamed Blackbird, in early 1996 (Bank, 1995). Since many synergetic effects have hit the Internet and enterprise technology the last years the development of the technology has accelerated. Examples of appearances at the market that have created synergic effects are: applets, thousands of Java technology-oriented start-ups, over a thousand books on Java technology, Netscape Communicator, thousands of Internet service providers, 60 million Internet users, 56K and cable modems, electronic commerce and commitments from major players such as IBM. The synergetic effects have also helped the Java platform to become adopted quickly in the industry. (Byouns, 1998) Any Java application can be downloaded over any network and installed without operating system or hardware platform compatibility issues. (http://java.sun.com/java2/whatis/, 2002-09-26) It was during another group meeting at Sun another question maybe popped up: “Why not mobile devices?” One reason why Java is a good language to develop is that many software developers know the Java language well and the increase of bandwidth in mobile networks will enable new mobile services. Sun therefore developed a Java version for mobile devices. In fact, today, September 2002, there are a number of mobile phones containing the Java platform on the market.

2.1.2 J2ME (Java 2 Micro Edition)

Because of limited resources in mobile devices in terms of limited displays for instance it is not possible to install the entire J2SE (Java 2 Standard Edition) Application Programming Interface (API) on a small device. That is the reason why “Micro Edition” was developed.(Muchow,2002) The idea with J2ME is to enable the customer to download applications to the mobile phone and to increase customer value with these applications.

J2ME stands for Java™ Platform, Micro Edition and is a very small Java application environment. It can be configured for a variety of market segments. (http://java.sun.com/j2me/j2me-ds-0201.pdf, 2002-09-26) However, J2ME is aimed at consumer devices with limited horsepower. Many of these devices (e.g. mobile phones and PDAs) do not have the option of downloading and installing beyond what was configured during the manufacturing process. With J2ME the small devices no more need to be “static” in nature.

A configuration comprises a virtual machine, core libraries, classes and APIs. Currently, there exist two different configurations: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC). CLDC is used for devices with constrained CPU (Central Processor Unit) and memory resources, such as mobile phones and PDAs. CDC is designed for the next-generation devices with more robust resources. The advantages with J2ME are that it will be simple for the customers to use, simple for the programmers to develop and also simple for service providers to deploy. (http://java.sun.com/j2me/j2me-ds-0201.pdf, 2002-09-26)

Examples of downloadable applications can be interactive games, life management tools, travel-related applications and information tools. Applications could for instance have a zoomable map needed on a business trip and when the application no longer is needed, the user can simply delete it from the memory.
(http://www.nokia.com/java/for_developers.html, 2002-10-15)

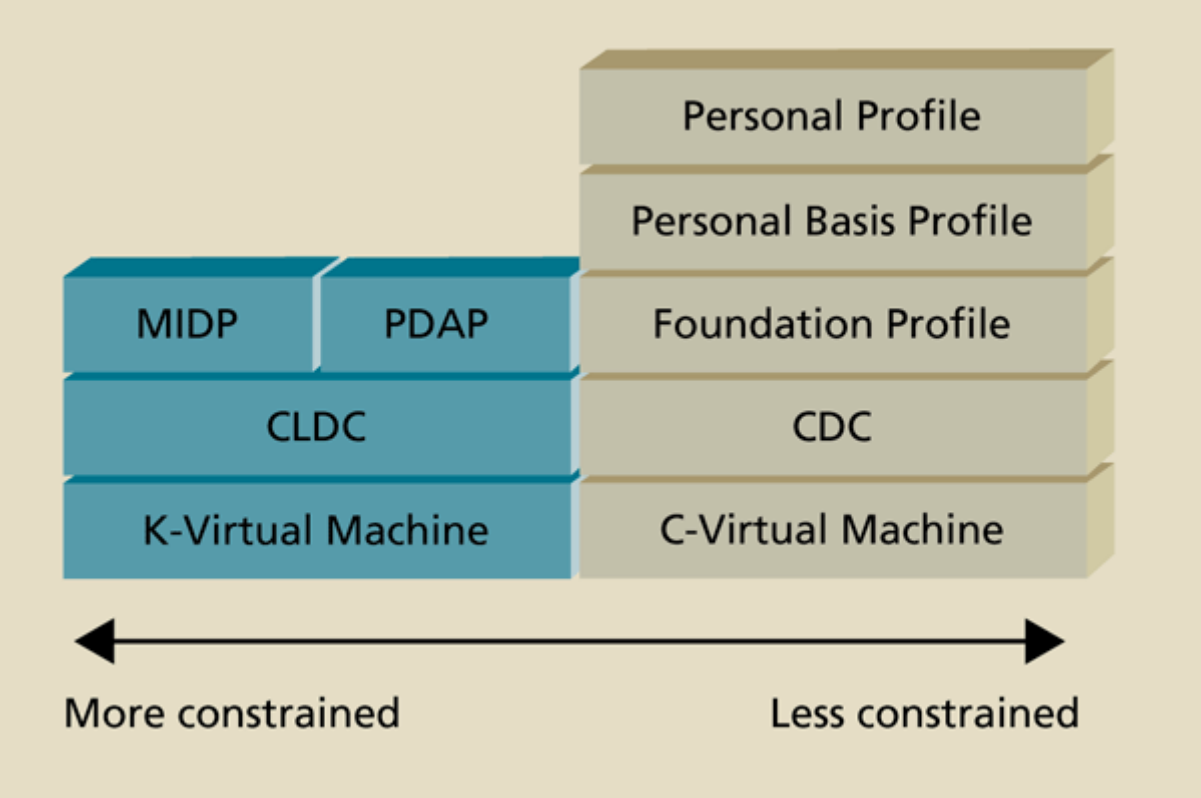

Figure 3. Difference between CLDC and CDC and virtual machines
(http://www.fawcette.com/javapro/2002_12/magazine/features/dhemphill/default_pf.asp)

CDC (Connected Device Configuration)
CDC is designed for the next-generation devices with more robust resources. These devices typically run on a 32-bit CPU and have 2 Mbytes or more memory available for the Java platform and its applications. (http://java.sun.com/j2me/j2me-ds-0201.pdf, 2002-09-26)

The characteristics of the devices that use CDC are:

- 512 kilobytes (minimum) memory for running Java
- 256 kilobytes (minimum) for runtime memory allocation
- Network connectivity, possibly persistent and high bandwidth

(Muchow, 2002)

CLDC (Connected Limited Device Configuration)
CLDC is the foundation of the Java runtime environment targeting small, resource-constrained devices, such as mobile phones and PDAs. To support the CLDC there are two virtual machines that can be used, the K Virtual Machine (KVM) and the CLDC HotSpot virtual machine. K is referring to kilobyte because the virtual machine only demands a few tens of kilobyte of memory when executing. The KVM is designed with the constraints of inexpensive mobile devices in mind. CLDC with KVM is suitable for devices with 16/32-bit RISC/CISC microprocessors/controllers. It only has 160 KB of total memory available for the Java technology stack. Of these 160KB, 128KB is used for the storage of actual virtual machine and libraries and the remainder is for Java applications. The CLDC HotSpot virtual machine is targeted for mobile devices with bigger resources. The implementation of the latter virtual machine is suitable for devices with 32-bit RISC/CISC microprocessors/controllers and with 512kB to 1MB of total memory available for the Java stack, including applications.
(http://java.sun.com/products/cldc/, 2002-09-26)

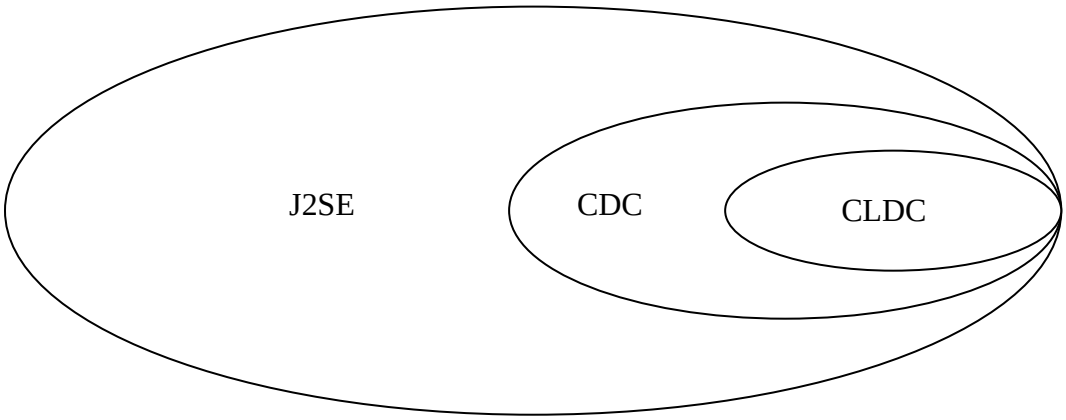

Figure 4. This figure represents how the different configurations are inherited from each other. (J2SE - Java 2 Standard Edition)

At the wireless market the utilizing of CLDC can retain a very small footprint, consume little power and provide as much capability as is needed for handheld devices.
(http://java.sun.com/j2me/j2me-ds-0201.pdf, 2002-09-26)

The characteristics of the devices that use CLDC are:

- 128 kilobytes memory for running Java
- 32 kilobytes memory for runtime memory allocation
- Restricted user interface

- Low power, typically battery powered
- Network connectivity, typically wireless, with low bandwidth and intermittent access

(Muchow, 2002)

2.1.3 MIDP 1.0, 2.0, CDC and CLDC comparison

The picture below illustrates the different parts of Java, and in which manner they are related.

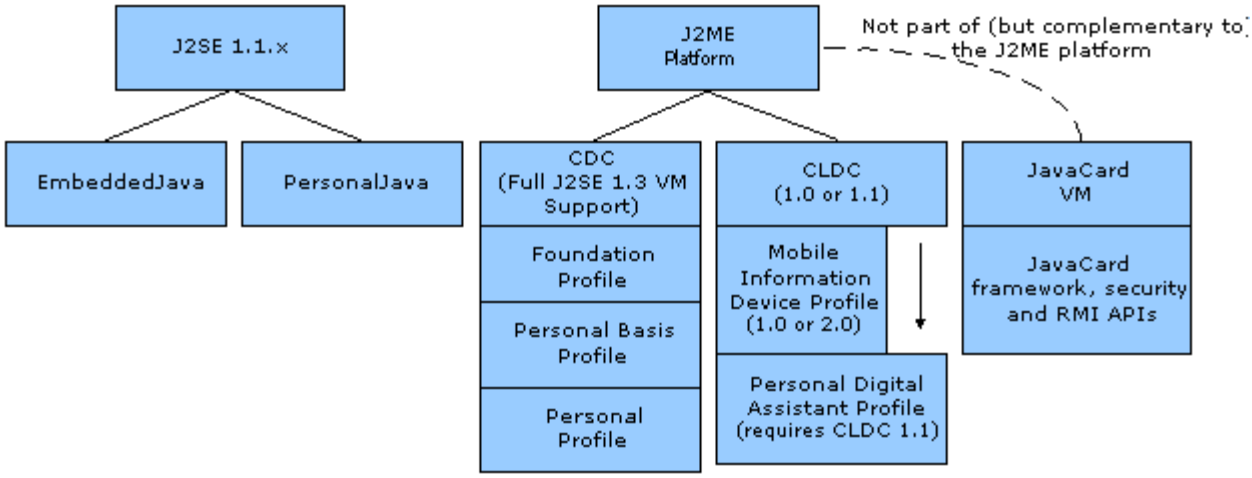

Figure 5. This show from what source the different configurations and profiles have sprung. (http://wireless.java.sun.com/getstart/articles/survey/survey_fig1.gif, 2003-01-30)

2.1.4 MIDP (Mobile Information Device Profile)

Mobile Information Device Profile, MIDP, is a part of the J2ME and defines the Java™ application environment for mobile information devices (MIDs). *MIDlet* is an application that runs in the MIDP environment. If one or more MIDlets are packaged together for distribution they are called a *MIDlet suite.*

To show the similarities and differences between MIDP 1.0, MIDP 2.0, CDC 1.0, CLDC 1.0 and CLDC 1.1 I have included a table as appendix. It contains a list of the Core Java Packages and it shows which packages are included in which configuration/profile.

Games

Stock information

Figure 6. Examples of J2ME applications (http://www.sumea.com/6_0/games.html, 2003-01-30; http://e-www.motorola.com/webapp/sps/site/application.jsp?nodeId=03M0ym4sDZxM0zbxntsZ , 2003-01-30)

2.1.5 Restrictions within J2ME

Java language differences
There are three main areas that distinguish the Java language for CLDC versus the Java Language Specification.

Floating point math is intrinsic processor intensive. It is also a fact that the majority of devices will not have special hardware for handling float point numbers. Therefore the CLDC implementation of the Java language does not support floats. This lack of support is carried throughout all Java code one writes. One cannot use float variables, constants, arrays and arithmetic or return values from methods.

Within a J2SE class, one can declare a method with the name finalize(). Finalization is a method that will be called before the garbage collector frees the object. Although the garbage collector frees memory used by an object, it is not aware of system resources that may have been acquired. The finalize() method is where one can place code to clean up allocated resources. Unfortunately, the finalize method is not supported in the CLDC. The reason is to find in the specification, but it is a bit sketchy. However, one can make a safe assumption and that is that the overhead and/or processor requirements are too steep.

The KVM (K Virtual Machine) supports a limited set of error handling exceptions. There are two reasons for that:

- Since the exception handling in J2SE is quite comprehensive it takes a significant demand on the system, which the CLDC cannot meet.

- Often times, embedded systems provide their own internal error handling. When the problem has come so far it's too late for the developer to deal with it. A simple solution for the most serious errors is to shut off the device. However, this way of solving the problem is definitely not convenient for the user and is more appropriate in a hardware point of view.

Java Virtual Machine Differences

The K Virtual Machine, KVM, differs from the Java Language Specification as follows:

Floating Point Math

The implementation of the Java language for CLDC lacks support for floating point numbers, as mentioned above.

Java Native Interface

In order to meet the goals of reduced memory requirements and to reduce the potential of corruption to system level information support for invoking native methods and APIs of other programming languages was eliminated. However, a KVM-implementation may link native code directly into the virtual machine. Unfortunately, an application that accesses the native code may not be portable to other KVMs.

Custom Class Loader

The VM is required by the CLDC to implement a class loader. This loader is being controlled tightly – it can't be replaced, overridden or modified. It is device-dependent, i.e. defined and implemented by the device manufacturer, and it includes how classes are loaded and in what manner error conditions are handled.

Reflection

One can use reflection in J2SE to get information about the running VM. This may include information about the class files loaded and their methods and fields. Unfortunately, reflection is not included in the KVM.

Thread Groups

For Thread Groups implementation, threads are processed on an object-by-object basis. The Thread Group class is not supported in the KVM. Thus, one cannot perform operations such as starting/stopping a group of threads through one method call. To solve this problem and mimic the functionality one can roll one's own thread processing code, e.g. by storing a group of threads objects within a collection class and to provide methods to start/stop all objects within the collection.

Finalization
The implementation of the Java language for CLDC lacks support for finalization, as mentioned above. This is also a fact for the KVM.

Weak Reference
The so-called Weak Reference to objects is allowed in J2SE. This means that the garbage collector recognizes that an object is being referenced and the object is still a candidate for garbage collection. Weak References is not supported in the KVM.

Handling security
All devices that are running a Java application need to be protected from malicious code (intentional or otherwise) that may access system information or resources. Regardless of the JVM, this type of "low-level" security is implemented with the use of class file verification. The JVM verifies all classes in class loaders and ensures that applications do not perform any dangerous operations. Since the verification at runtime is computationally expensive for MIDP VMs, MIDP uses a two-stage bytecode verification model. MIDP VMs do not perform complete bytecode verification at runtime. Instead the application developers have to run a pre-verification of the classes on a development platform before deploying the application. The pre-verification process optimizes the execution flows, creates stackmaps containing catalogs of instructions in the application, and then adds the stackmaps to the pre-verified class files. At runtime, the MIDP VM does a quick linear scan of the bytecode, matching each valid instruction with a proper stackmap entry. Since MIDP lacks a complete security model some J2SE features are disabled in J2ME that otherwise would pose a security risk. As an example it is a fact that user-defined class-loaders are not allowed, to prevent illegal overloading of core classes.
(http://www-106.ibm.com/developerworks/webservices/library/wi-secj2me.html?dwzone=webservices, 2003-01-29)

2.1.6 Restrictions within MIDP
LCDUI (Liquid Crystal Display User Interface) does not provide overlapping windows for MIDP.

Unfortunately, some APIs are not included within MIDP, but included by specific mobile phone producers. Siemens have APIs for calling, SMS and phonebook (restricted). Nokia has an extra API supporting SMS for their phones. Motorola has an UI API and an API for Bluetooth. Only the corresponding manufacturer supports the extra API. Ericsson and Samsung do not have any extra APIs at all.

2.1.7 MIDP 1.0
MIDP 1.0 is the first version of MIDP and therefore it does not supply the developers with all desired API's. It provides standard APIs for application lifecycle, HTTP network connectivity, user interface, and persistent storage. However, SMS (Short Message Service), CBS (Cell Broadcast Service) and USSD (Unstructured Supplementary Service Data) are not supported. Cell Broadcast permits unacknowledged text messages to be broadcast to all receivers within a particular region (http://www.etsi.org/, 2003-01-08). USSD is a type of signalling.

2.1.8 MIDP 2.0

Specification process

During the specification process indications were given that USSD was to be supported in MIDP 2.0 or in the JSR 118 (Java Specification Request), also known as WMA (Wireless Messaging API). WMA includes APIs for SMS and CBS and the mobile phone manufacturers choose to support it or not in the phones. Actually, in the first draft of WMA, USSD was supported. However, support for USSD was finally rejected both in MIDP 2.0 and in the WMA-package.

The final draft of MIDP 2.0 was published in November 2002 and includes many enhancements and additions to MIDP 1.0. Estimates in December 2002 are that MIDP 2.0 devices will be available in large volumes in summer 2003. The enhancements included cover the following areas:

- Secure networking
- Multimedia
- Form Enhancements
- The Game API
- RGB Images
- Code Signing and Permissions

Secure Networking

MIDP 2.0 not only supplies HTTP connections, but also HTTPS connections. Basically, HTTPS is a HTTP over the Secure Sockets Layer (SSL), which is a socket protocol. It encrypts data sent over the network and provides authentication for the socket endpoints. MIDP 1.0 implementations can support HTTPS, but one cannot rely on its availability. What MIDP 2.0 does is to provide a stable, consistent foundation for wireless applications that deal with money or sensitive information. The CLDC's Generic Connection Framework in the *javax.microedition.io* package provides HTTPS support. The two following examples show how easy it is to construct a HTTP and a HTTPS connection:

HTTP connection:

String url =”http://www.cert.org/”;
HttpConnection hc = (Httpconnection)Connector.open(url);

HTTPS connection:

String url =”https://www.cert.org/”;
HttpsConnection hc = (Httpsconnection)Connector.open(url);

“Javax.microedition.io.SecurityInfo” can be used to retrieve the certificate, protocol, version, and cipher suite used by a HTTP connection, and “javax.microedition.pki.Certificate” represents a cryptographic certificate. They are both included in MIDP 2.0.

Multimedia

MIDP 2.0 has mobile media APIs that support playback of sampled audio. Implementations using these APIs must be able to play WAV files and are free to support additional audio formats.

There have been some changes in *javax.microedition.lcdui*. The form layout is considerably more sophisticated than it is in MIDP 1.0. Items are laid out like ordinary text left to right in rows that stack from top to bottom and one can modify this layout.

The Game API
Many of the enhancements for game developers can be found in *javax.microedition.lcdui.game.* There are five classes in the Game API that extends MIDP's graphics capabilities. Contents of the screen can be composed of different layers where one layer could for example be the background. A second layer could be a spaceship and a third a clouds, rain, etc.

RGB Images
With the MIDP 2.0 it is possible to represent an image as integer arrays, which allows MIDlets to manipulate image data directly.

(http://wireless.java.sun.com/midp/articles/midp20/, 2003-01-08)

2.1.9 Java Servlet Technology
Java Servlet Technlogy enables web developers to extend the functionality of a web server. A servlet can be thought of as an applet that runs on the server side and therefore servlets have made many web applications possible. Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of the CGI (Common Gateway Interface) programs (similar as servlets). (http://java.sun.com/products/servlet/, 2002-11-05)

A servlet is a file written in Java, which then is compiled with for example Javac, which is a Java compiler. When the class (the servlet) is called from a file (HTML, JSP or WML) it is run through the JVM loaded on the server. One needs servlets to be able to build dynamic web pages and connect mobile phones to databases. One can use CGI instead of servlets but since servlets have several advantages (listed below) they are preferred.

- **Effectiveness.** With regular CGI a new process is started for every HTTP request, and if the CGI program itself is small and fast the majority of the consumed time might be starting the process. With servlets the JVM stays up and each request is handled by a lightweight Java thread, which is much more efficient while one doesn't have to start a new session for each thread. Another advantage is that if there are *n* simultaneous request to the same CGI program, the code of the CGI program is loaded into memory *n* times. With a servlet there are *n* threads but only one servlet class loaded into the memory.

- **Powerfulness.** Servlets can do things that are difficult or maybe impossible with CGI. They can talk directly to a web server and thereby the process of getting pictures and other stored data is simplified. They can share data between them, which makes it easier to implement database connection pools. They can maintain information from request to request, simplifying session tracking and caching of previous computations.

- **Portability.** Since servlets are written in Java they are supported on basically every major web server, generally without any changes made to the code.

2.1.10 JSP (Java Server Pages Technology)

Java Server Pages (JSP) technology allows one to create web content easily from both static and dynamic components. JSP technology supports all the dynamic capabilities of Java Servlet technology but provides a more natural approach to creating static content. The main features of JSP technology are:

- A language for developing JSP pages, which are text-based documents and describe how to process a request and construct a response
- Constructs for accessing server-side objects
- Mechanisms for defining extensions to the JSP language

A *JSP page* is a text-based document that contains two types of text: Static template data, which can be expressed in any text based format, such as HTML SVG, WML and XML and JSP elements, which construct dynamic contents. A JSP page handles requests as a servlet. Thus, the life cycle and many of the capabilities of JSP pages (in particular the dynamic aspects) are determined by the Java Servlet technology. When a request is mapped to a JSP, it is first handled by a special servlet that checks whether the JSP page's servlet is older than the JSP page. If it is older, it translates the JSP page into a servlet class and compiles the class. One of the advantages of JSP pages over servlets is that the build process is performed automatically.
(http://java.sun.com/webservices/docs/1.0/tutorial/doc/JSPIntro.html, 2002-11-05)

2.1.11 Optional Packages

In the early development of J2ME the focus was on developing configurations and profiles but as time passed they realized that something else was needed, Optional Packages. Optional Packages are similar to profiles also a set of APIs but unlike profiles they do not define a complete application environment. An optional package is always used in conjunction with a configuration or a profile. The optional packages give developers access to additional functions of the devices and since these specific capabilities are not universal for all devices, the manufacturers can choose to preload whichever profiles and optional packages they see fit to their devices. Note that a package normally has to be preloaded onto a device, as part of the runtime environment, by the manufacturer, for developers to be able to use them. If one would try to use an optional package without it being preloaded, it simply wouldn't work; there would be no support for the classes. Because just like profiles and configurations, optional packages are specified through the Java Community Process, each has its own reference implementation (RI) and test compatibility toolkit (TCK). Besides aiding the vendors in implementing optional packages as part of their runtime environments, the RI and TCK also ensure that those implementations are done consistently and correctly, no matter which device that is being used. It usually takes some time after an optional package specification has been finalized before implementation appear on commercial devices.
Some examples of optional packages are: JSR 66: RMI Optional Package, JSR 120: Wireless Messaging API, JSR 135: Mobile Media API, JSR 169: JDBC for CDC/FP.
(http://wireless.java.sun.com/midp/articles/optional/ 2002-12-18)

Bluetooth
Bluetooth is an important emerging standard for wireless integration of small devices. The specification is labelled JSR82, and it is now in JCP Final Release Stage. The JSR82 specification will standardize a set of Java APIs to allow the Java-enabled devices to integrate into a Bluetooth environment.

The specification will include basic support for, at least, the following Bluetooth protocols: RFCOMM, OBEX, and Service Discovery protocols. Additional protocol support may be added in future versions. The specification is primarily targeted at native Bluetooth protocols.

The Java APIs for Bluetooth are targeted at devices characterized as follows:

- 512 K minimum total memory available (ROM/Flash and RAM). Application memory requirements are additional.
- Bluetooth network connection.
- Compliant implementation of the J2ME Connected Limited Device Configuration.

The specification will define the APIs such that it will be extensible to other Bluetooth protocols which exist today (i.e. Home RF), or that might come about in the future. In addition, the APIs will be specified in a way to allow layering for more capable Java platforms such as the CDC, J2SE, and J2EE. (http://www.jcp.org/en/jsr/detail?id=82, 2003-01-20)

2.1.12 JSR (Java Specification Request)

JSR stands for Java Specification Request and they are the actual descriptions of proposed and final specifications for the Java platform. There are six stages a proposal has to go through:

1. JSR Review - A specification is initiated by community members and approved for development by the Executive Committee
2. Community Review - Once a JSR is approved, a group of experts is formed to develop a first draft of the specification that both the community and the Executive Committee review.
3. Public Review - The JSR draft goes out for review by the public where anyone with an Internet connection can read and comment on the draft. The Expert Group uses the public feedback to further revise the document.
4. Proposed Final Draft - The version of the draft specification that will be used as the basis for the RI and TCK.
5. Final Release - The leader of the Expert Group then sees that the reference implementation and its associated Technology Compatibility Kit are completed before sending the specification to the Executive Committee for final approval.
6. Maintenance Review - The completed specification, reference implementation, and Technology Compatibility Kit are updated in response to ongoing requests for clarification, interpretation, enhancements, and revisions.

(http://www.jcp.org/en/jsr/stage, 2002-11-20)

To point out how long time it takes for a proposal to go through this chain, I will use JSR82 (bluetooth) as an example. From start to finish, JSR 82 took nearly one and a

half years to finish. The Executive Committee approved its proposal October 2000. The original 14-member expert group took shape the next month, and seven additional members were appointed in July 2001. The community reviewed the specification in November, with public review following in December, and the final draft was proposed in January. On March 4, 2002, the Executive Committee approved the final specification. (http://www.jcp.org/en/press/success/bluetooth, 2003-01-20)

2.1.13 Smart cards

A smart card is a card with embedded chips, either a microprocessor and a memory chip or only a memory chip with a non-programmable logic. The microprocessor card can add, delete, and otherwise manipulate information on the card, e.g. a Java program. A memory-chip card (for example, pre-paid phone cards) can only undertake a pre-defined operation.
Smart cards, unlike magnetic stripe cards, can carry all necessary functions and information on the card, e.g. for a transaction. Therefore, they don't require access to remote databases at the time of the transaction.

2.1.14 Java Card™ Technology

Java Card™ Technology enables smart cards and other devices with limited memory resources to run small applications, called applets, which utilize Java™ Technology. Java Card Technology offers platform independence, the ability to store and update multiple applications, and compatibility with existing smart card standards.
A complementary technology is Java 2 Platform, Micro Edition. The two technologies make it easy to integrate essential consumer technology into a complete Java software solution. New applications and services can be rapidly and securely built, tested, and deployed. This reduces development costs, adds product differentiation and enhances value-add for customers.

Industries Embracing the Java Card
Most smart cards can be fitted with Java Card technology, including:

- SIM cards used in cell phones on most second and third generation wireless networks
- Financial cards providing both online and offline transactions
- Government ID cards

On the majority of cellular telephone networks, smart cards (SIM cards) are required to activate the telephone. The SIM card authenticates the user and provides encryption keys for digital voice transmission. If fitted with Java Card technology, SIM cards can also provide transactional services such as remote banking and ticketing.

In the banking industry, smart cards can give users secure access to a wide array of networked financial services including cash machines, bill paying, and bridge tolls. Java Card-based smart cards can host multiple financial applications in a single smart card and deliver third-party services such as mileage programs or secure, on-line trading.

A wide variety of other applications are available wherever security and authenticated identity are important. Examples of services are providing access to facilities and medical records.

Java Card technology will enhance consumer access to new, e-commerce services through a number of connected appliances. Examples of markets where the majority of products now available include built-in smart cards are cellular phones and pay-TV equipment.

Advantages of Developing with Java Card
The Java Card technology offers all the advantages of developing applications in the Java programming language:

- High programmer productivity
- Object-oriented programming with greater code modularity and reusability
- Java language protections apply to Java Card applets, enforcing strong typing and protection attributes
- Availability of powerful off-the-shelf development tools

Components of Java Card Technology
The Java Card technology consists of the Java Card specifications and the Java Card Development Kit, which includes a reference implementation based on those specifications.
Providing the basis for cross-platform and cross-vendor applet interoperability, the Java Card specifications are as follows:

- Java Card 2.2 Virtual Machine Specification, Revision 1.0, Sun Microsystems, Inc.
- Java Card 2.2 Runtime Environment (JCRE) Specification, Revision 1.0, Sun Microsystems, Inc.
- Java Card 2.2 API Specification, Revision 1.1, Sun Microsystems, Inc.

Java Card Virtual Machine (JCVM) Specification
The Java Card 2.2 Virtual Machine Specification defines the features, services, and behaviour that are required of an implementation of the Java Card technology. It includes the instruction set of a Java Card virtual machine and the supported subset of the Java language. It also includes file formats used for installing applets and libraries into devices, like smart cards, which implement Java Card technology.

Java Card Runtime Environment (JCRE) Specification
The JCRE 2.2 Specification complements the Java Card 2.2 API Specification and defines the necessary behaviour of the runtime environment in any implementation of the Java Card technology. An implementation of the Java Card Virtual Machine is included in such implementation. The Java Card Application Programming Interface (API) classes, and runtime supports services such as the selection of applets are also included.

Java Card API Specification
The Java Card API is compatible with formal international standards and industry-specific standards and it contains the class definitions required to support the JCVM and the JCRE.

Java Card Development Kit
The Java Card Development Kit is a suite of tools for designing Java Card technology-

based implementations and developing applets based on the Java Card API Specification. It consists of:

- The C-JCRE is a reference implementation of the Java Card Runtime Environment written in the C programming language. The C-JCRE also includes the Java Card Virtual Machine (JCVM) interpreter.
- Off-card platform components such as the Java Card Converter and the Java Card Verifier complement the C-JCRE to provide a complete development chain.
- Additional design and testing tools enable developers to prototype and test Java Card applications.

Java Card Technology Licensees
Java Card technology is licensed to smart card manufacturers representing more than 90 percent of the world's smart card manufacturing market. The platform that is defined by the Java Card technology provides a range of new opportunities for original equipment manufacturers (OEMs) and their partners across multiple industries. The GSM (Global System for Mobile communications) cellular telephone industry is an example of a large, global market for smart cards based on Java Card technology. Major financial institutions are deploying Java Card technology to provide a rich set of applications in a single card.

(http://java.sun.com/products/javacard/datasheet.html/, 2002-11-13)

2.1.15 The Java Card API
The APIs included for security are:

- AES Key
- DES Key (Triple DES supported)
- DAS Key
- EC Key
- RSA Private Key
- RSA Public Key

Therefore, very sophisticated cryptographic solutions for secure mobile connections can be obtained with the Java Card Technology.

2.1.16 Java phones in other countries, distribution of earnings
In Japan Java phones have been available on the market the last couple of years. The Japanese mobile operator NTT Docomo's revenues from customers with Java phones are more than double compared to the other customers. In Korea the mobile operator LG Telecom makes approximately 50 SEK per month more on customers with Java enable phones than on the other customers. They are doing this even though the Java programs are very simple. According to Rikard Kjellberg, Technology Manager at Ellipsus AB, one of the reasons is that Japanese customers appreciate a figure just jumping up and down on the display, but the customers in Europe are not willing to pay for that. (http://www.nyteknik.se/pub/ipsart.asp?art_id=20050, 2002-12-16)

2.2 Mobile Network

The different parts of an IN (Intelligent network), the GSM network (Global System for Mobile communications) and the GPRS network (General Packet Radio Service) are presented in figure 7.

AUC	Authentication Centre
BSC	Base Station Controller
BTS	Base Transceiver Station
EIR	Equipment Identity Register
GMSC	Gateway MSC
GGSN	Gateway GPRS Support Node
HLR	Home Location Register
IN	Intelligent Network
MS	Mobile Station
MSC	Mobile Services Switching Centre
PCU	Packet Control Unit
SGSN	Serving GPRS Support Node
VLR	Visitor Location Register

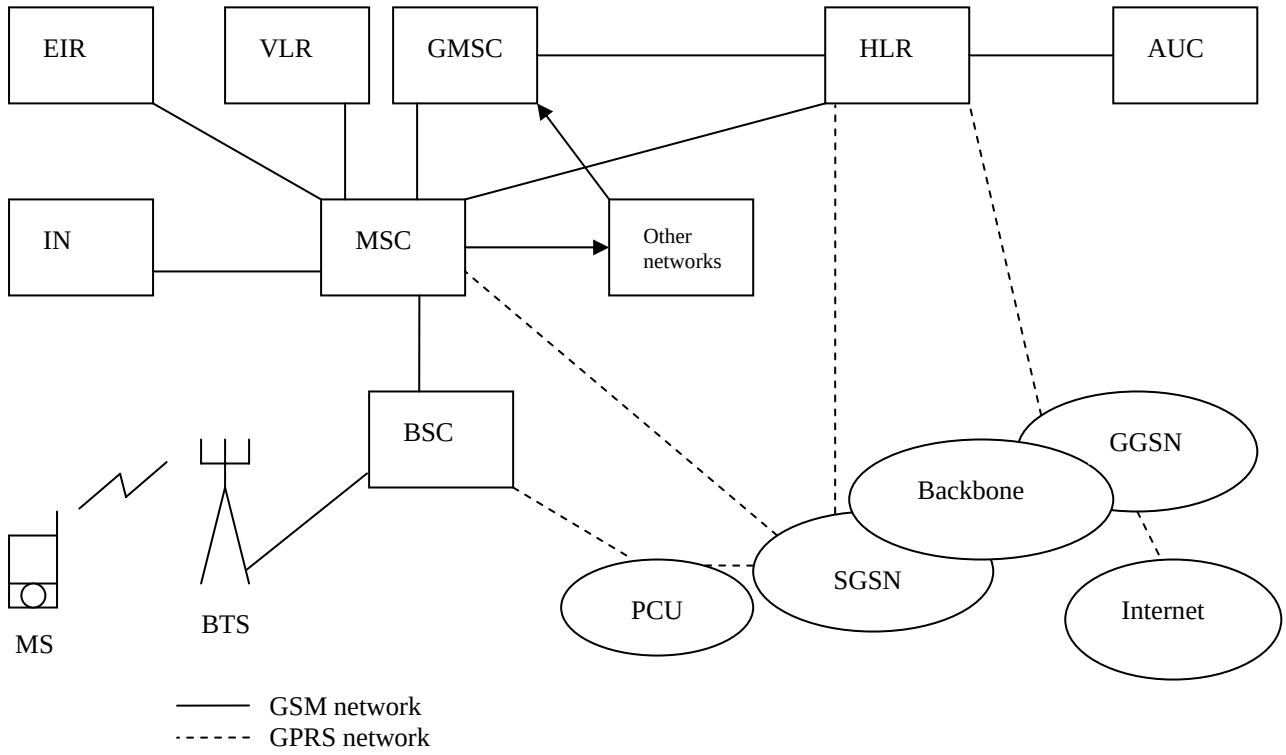

Figure 7. This is a modified version from GSM System Overview, APIS Training & Seminars and GPRS Protocols and Procedures, Enea Epact.

2.2.1 GSM network with an IN node

Mobile Station (MS)

The MS is nothing else but the SIM, “Subscriber Identity Module”, and the ”Mobile Equipment” (ME). The SIM is a separate physical entity that contains all information regarding the subscription. It is an IC-card, or a “smart card”. The Mobile Equipment is the actual piece of hardware enabling radio communication with the system. Without a SIM the Mobile Equipment cannot be used in the GSM network, except for emergency

calls. The SIM can be put into different mobile equipment and then all calls are routed to that mobile equipment.

Base Station Systems

Base Transceiver Station (BTS)
This is the radio part of the system. The main task is communication on radio with the MS:s. Each BTS covers a cell with transmitted radio waves. The BTS contains all radio equipment necessary to stay in touch with the MS.

Base Station Controller (BSC)
The BSC controls and supervises all underlying BTS:s. The BTS takes care of the actual radio communication and the BSC is in charge of all action that is taken. That means that the BSC tells the BTS:s what to do, when to transmit, what power to use, etc.

Switching System

Mobile services Switching Centre (MSC)
The MSC sets up, supervises and releases calls. It can connect calls within the GSM network, between a mobile subscriber and a subscriber in the PSTN (Public Switched Telephone Network) or in other networks.

IN
IN (Intelligent network) is, according to ETSI (European Telecommunications Standards Institute), an architect concept for all telecommunication networks, which aims to ease the introduction of additional services. An IN service is for example pre-paid, Dirigent Centrex or Mobile Switcher. A number is first dialled on the MS. When the send-button is pressed, the call will be set up, via the BTS, onwards to the BSC and finally to the MSC. The MSC then analyses the digits dialled and sets up the call to wherever it is to go. If the subscriber is having an IN service the call is then transmitted to the IN node. The IN node then executes the call request depending on which type of subscription that is calling. If a Pre-paid customer is calling the IN node checks whether the subscriber has enough credits to make a phone call before the call is being set up. If the customer is a Dirigent subscriber and uses an extension number within his/her groups, then the IN node translates for instance the extension number 52 23 into the mobile public number 070 684 52 23. After the translation the call is being forwarded in the GSM network. In this case it would be forwarded to the GMSC.

Gateway Mobile services Switching Centre (GMSC)
It is necessary to have a gateway between GSM and other networks. A call to a subscriber in the GSM network means that the call will first be routed to the GMSC. The responsibility of the GMSC is then to find out what part of the GSM network the MS is situated by questioning HLR (Home Location Register) and also for routing the call there.

Home Location Register (HLR)
The HLR is a global database that keeps track of where the subscriber approximately is. Normally, there is just one HLR per operator. However, this is not the case with some operators. In the HLR data about all subscribers belonging to the particular network are

stored. Among the data, the information in which MSC/VLR service area the MS is will be stored.

Visitor Location Register (VLR)
The VLR is a regional database that is linked together with every MSC. This database stores information about all the subscribers that are registered in the corresponding MSC-service area at the moment. When the HLR has informed the GMSC about in which MSC/VLR service area the subscriber is, a more detailed description of in which location area the MS will be found in the VLR.

Authentication Centre (AUC)
The AUC is a database that deals with security. It produces keys for authentication and encryption and for this it contains parameters for all the subscribers belonging to that operator.

All MS:s can be asked to go through an authentication procedure before they are allowed to do anything else in the system. All necessary details for authentication are found on the SIM-card.

Equipment Identity Register (EIR)
The EIR database stores information about the pieces of mobile equipment (i.e. the hardware). It can keep track of information such as stolen equipment and non-type approved equipment, etc. The AUC is concerned with the SIM, but the EIR is concerned with the mobile equipment.

(GSM System Overview, APIS Training & Seminars)

2.2.2 GPRS (General Packet Radio Service)

GPRS is a packet switched network and the user can select different external networks. Multiple users share common resources, which lead to an effective use of the bandwidth. The users can always be online and radio resources are being allocated when needed. The customer can subscribe for different levels of QoS, Quality of Service. It's even possible to use different QoS levels for different applications. The charging is volume based and can depend on the QoS. What is called GPRS network in figure 2 is basically just an "add on" to the GSM System. The complete GPRS network also includes a MS, BSS, MSC/VLR, HLR and a GMSC.

PCU – Packet Control Unit
The Packet Control Unit can be placed between the BSC (Base Station Controller) and the SGSN, (Serving GPRS Support Node), by the BSC or by the SGSN. It is responsible for the GPRS packet data radio resource management in BSS (Base station systems). It is a standard interface to the SGSN and it requires new hardware in the BSC.

SGSN – Serving GPRS Support Node
The SGSN forwards IP packets addressed to/from a mobile station. It is managing the mobility, which means that it keeps track of the mobile stations within the service area and handle registration of a mobile station – attach. The SGSN also compresses and encryptions data and collects charging information related to the usage of the air interface.

GGSN – Gateway GPRS Support Node
The GGSN is an interface for external packet data networks (e.g. Internet). It works as a session manager, which means that it keeps track of ongoing data transfers to/from mobile stations and assigning an IP address to each session – activation. The GGSN is also collecting charging information related to the usage of external network resources. The GGSN looks like an ordinary router on the Internet.

(GPRS Protocols and Procedures, Enea Epact)

2.2.3 UMTS (Universal Mobile Telecommunications System)
The UMTS is being built using the same principles as the GPRS. However, UMTS uses a new radio system instead of GSM.

2.2.4 USSD (Unstructured Supplementary Service Data)
USSD (Unstructured Supplementary Service Data) is a signalling type that is used i.e. for mobile extension and balance information for pre-paid customers. When signalling USSD one uses the first time slot among the eight time slots in the GSM technology. It is being used by SMS, USSD and as a control slot. When a new call sets up it first checks with the control in the first slot whether one of slot 2-8 is free. USSD is not supported in neither MIDP 1.0 nor MIDP 2.0.

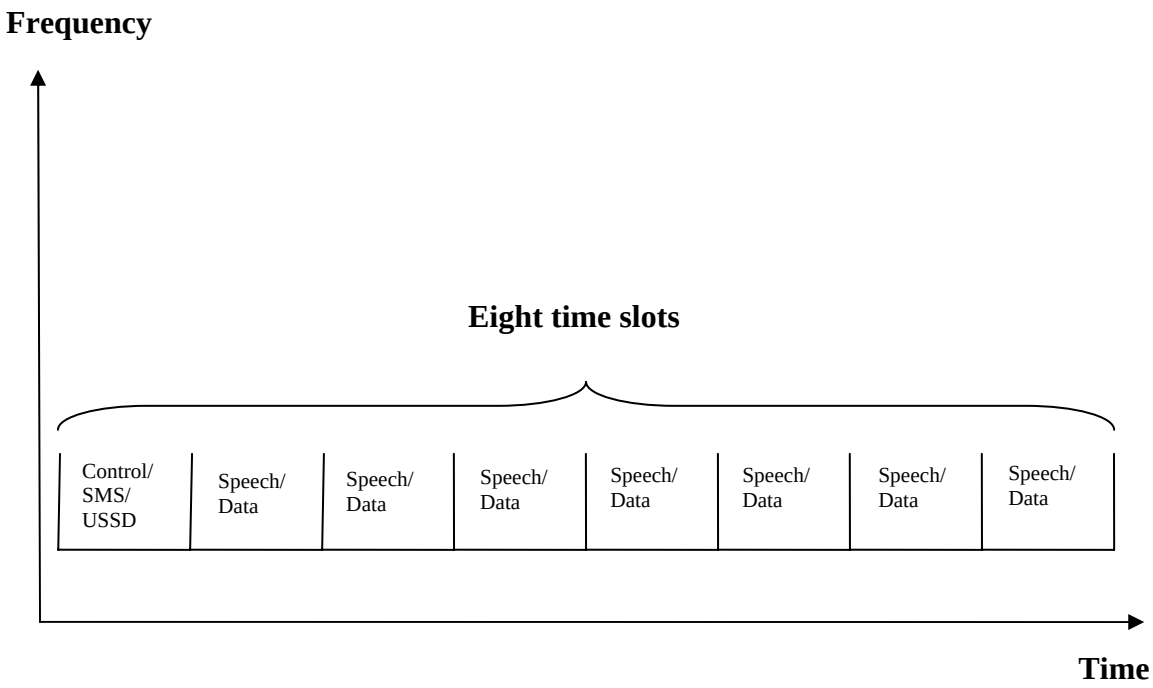

Figure 8. This shows how the time slots are used in the GSM network. I wanted to show this because I had an idea to use the USSD function instead of the SMS function to send messages when the network was overloaded, but since they both use the same timeslot it will not work as I had hoped.

2.3 Mobile business services

2.3.1 Dirigent

Dirigent is a product family, which includes the services Dirigent, Centrex and Mobile Switcher. These services are all IN services. The idea with the service Dirigent is to give the customer an integration of the switchboard and the companies' mobile phones. Dirigent is implemented in an IN platform (Intelligent Network). One feature of the service is called "mobile reference". If the customer activates a mobile reference another person that is calling the customer gets a message telling him/her that the person with the dialled number is having lunch and will be back 12:30, for instance. However, the mobile phone must be shut off. Otherwise it will ring as usual. Here is a complete list of the mobile references available, including their dial codes (the prefix depends on the type of subscription the customer is having):

CODE	**FUNCTION NAME**	**DIAL CODE**	**UNIT**
00	Erase all	<prefix> 136 00	
0	Lunch	<prefix> 136 0 <unit>	hour/minute
1	Gone home	<prefix> 136 1	
2	Business errand	<prefix> 136 2 <unit>	hour/minute
3	Meeting	<prefix> 136 3 <unit>	hour/minute
4	Business trip	<prefix> 136 4 <unit>	day/month
5	Part time	<prefix> 136 5 <unit>	day/month
6	Vacation	<prefix> 136 6 <unit>	day/month
7	Client visit	<prefix> 136 7 <unit>	day/month
8	Temporarily out	<prefix> 136 8 <unit>	hour/minute
9	Course	<prefix> 136 9 <unit>	day/month

Table 1. Dial codes for Dirigent
(Tjänstebeskrivning "Dirigent", Telia Mobile AB)

2.3.2 Centrex

The idea of Centrex Mobil is that the operator takes over the whole responsibility of the telephone switcher. What Telia Mobile offers the customers is a more complete service than before, when the customers had to buy and support their own telephone switcher. Telia Mobile supports the product and that includes for instance automatic upgrades to the latest technology within network solutions. The technology also offers open user interfaces, which means an integration of the normal PBX (Private Branch Switchboard – telephone switcher) and the GSM solution. Centrex can offer the feature Mobile Reference. To activate the feature one has to press **990p*23*1#**, where "p" creates a pause for approximate three seconds. "p" is generated on an Ericsson phone by pressing "*" a while and with a Nokia one presses "*" three times. One can also activate a call forward with Centrex from the mobile device with the following sequence **990p*21*extension#**.

(Tjänstebeskrivning Mobil anslutning Direkt i Telia Centrex)

2.4 Viruses

2.4.1 Computer viruses

The number of yearly reports concerning computer viruses has increased approximately exponentially the latest years. However, the relative part of infection has decreased heavily with the increase of reports. During 2001 there were 112 different types of viruses reported and 22 types of these were new.

2.4.2 Mobile viruses

In Japan, the mobile phones are becoming so advanced that the first viruses specifically developed for them can be expected to appear soon. E-mails containing misleading links and attachments, that make the phones freeze have been a small problem. However, most problems have been due more to badly designed and annoying programs or attachments than to purposefully malicious attacks.

The Information-Technology Promotion Agency (IPA), in Japan, published the results of a test in March 2002. The test examined whether viruses developed for PC-based Java could damage Java equipped mobile phones. Unfortunately, the report is only available in Japanese at the following web page:
http://www.ipa.go.jp/security/fy13/report/mobile_security/java-keitai.pdf

(Embassy of Sweden, Science & Technology Office, Sweden-Japan IT Program, *Internet security issues*, Sabine Ehlers)

3 Empiry

In the empiry part the Java application for Dirigent being developed is presented. I also show by presenting some test results that the application makes the usage faster. The Centrex service is then treated and finally, empirical tests using USSD and Servlets by a Java application are presented

3.1 Mobile business services

3.1.1 Dirigent

It is quite complicated to activate a mobile reference since you have to know 11 different dial codes by heart. For example, if you want to notify that you are having lunch and you will be back at 12.30 you would dial “0013601230” (if the prefix is “00”, which it was with our test subscription). It can be quite difficult to remember all these codes but there is another way, and that is to call “<prefix> 136”. Then you follow the voice to set the mobile reference. However, this is a quite slow way of doing it and I have shown that it goes much faster with a Java application.

When I was reading the description of the service I realized that it must be very difficult to remember all the codes. Therefore I developed an application, with MSc student Daniel Hellstrand, which would do the job for the customer. By following menus it would be easier for the customer to activate features of the service.

I did some tests to measure the time calling 00136 followed by the instructions to activate a mobile reference. I also tested how long it took for the same person to activate the same feature with the corresponding settings with our Java application. I used two different categories of people, inexperienced and experienced, corresponding to a customer that is not very familiar with the service and one that is. When dialling, both the experienced and inexperienced user had to listen to the full length of the voice messages, since there are quite many options to remember. I used the first option “gone for lunch” and the second last option “temporary out” for the both behaviours.

The pictures below show how easy it is to use the interface.

Figure 9. When you decide that you want to use the program, the phone looks like this. You then press the button associated with Java, the left one.

Figure 10. When you have started the program this menu appears. It is quite self-explanatory and you simply make a selection which suites your needs at the time. If you are going for lunch you press “lunch", if you are going away on business you make that selection. As you can tell by the buttons you have two choices. Press ”Exit” to exit the program, press “Välj” to choose the active choice.

Figure 11. Suppose you choose that you are going to lunch. You will then get this screen, which asks you to enter the time of your return. Some of the choices will require that you enter the time of return, and some choices will require you to enter the date of your return. After you have entered the time you will press the button corresponding to the "Ok". If you want to exit you press the button corresponding to "Exit".

Figure 12. If you press "Ok" on the previous screen this screen will appear. The phone has detected that a Java program wishes to make a phone call and checks with you to make sure this is ok. This is a security precaution taken by the manufacturer to make sure Java programs are not making any phone calls it should not be. To confirm that the phone call can be made, press the button corresponding to "Ja", if it is not ok for the phone to make the call, press the button corresponding to "Nej". If you press "Ja" the phone will make a phone call and set the mobile reference you have chosen. When the signals are sent, you must hang up the phone, this is due to a limitation in the phone, since the Java program is terminated when the call is initiated.

I let some friends and acquaintances that have not used the service and do not use computers that often try to activate the feature. As experienced person I used some persons at the Service Design department at Telia Mobile. I measured the time it took for the persons to activate the features and then I calculated the average time. The following tables show how long time it took with the different alternatives.

BEHAVIOUR		
Inexperienced	Experienced	OPTION
54	35	Lunch
68	32	Temporary out

*Table 2. Testing Dirigent's mobile reference by **calling 00136 and follow the instructions***

BEHAVIOUR		
Inexperienced	Experienced	OPTION
30	23	Lunch
38	26	Temporary out

*Table 3. Testing Dirigent's mobile reference **using the Java program***

There are some obvious limits using Java at the moment. One of them is that when a phone initiates a phone call the Java program is terminated. The result of this is that there is no way of handling a return value from the contacted source.

Another thought I had was that if services could be controlled by either USSD or SMS you could make the controls better and faster. Thus, eliminating the need for the user to hang up the phone call after the redirection is set. I have been investigated the USSD issue to try to find out why it does not work. I tried to get a hold of the source code for the com.siemens.mp.gsm.* classes but it seems like these are not released by Siemens, which makes it impossible for me to check this issue any further.

After some more research and testing I have come to the conclusion that it is not possible to use USSD in a Java application on any cellular phone since the APIs do not support it. When the optional package WMA started emerging the specifications reported that USSD was to be included but this idea was dropped in an early stage. However, Siemens and Nokia are able to use SMS though.

3.1.2 Centrex

Another product that I have tried to improve with Java is Centrex. It is a product similar to Dirigent but it is larger and contains more functions, approximately 40, such as answering groups, phone conference (up to ten people) and of course redirection of calls depending on the cause of absence. To set a mobile reference for lunch in Centrex you would have to send these DTMF control digits: 990p*23*1# (where "p" is the character for "pause") by calling. Siemens is the only mobile phone manufacturer that supports calling with an extra API. Therefore Siemens is the only mobile phone that could work doing a mobile reference within Centrex. However, it turned out that it didn't work anyway. While Siemens' calling API neither supports "p" nor # or * it's

impossible to send a string containing these symbols from a J2ME program. I just took the code from our Dirigent application and changed it to the wished sequence for a mobile reference in Centrex.

The main problem using Java to do this is the same as in the USSD issue, only digits can be used when the outstring from a Java program is being sent to the calling function within the phone. In this case when I tried to use 566y4"p", "*" and "#" I immediately ran into problems. This means that the command cannot be carried out. This is caused by the limits in the additional API delivered be Siemens, which enables SMS and Call capabilities.

3.2 J2ME

3.2.1 USSD

I changed the output string in the MIDlet above into “*120#”, which is a balance request for the pre-pay service that Telia Mobile offers. This request is based on USSD but sending the new string did not work. This means that MIDP1.0 and/or Siemens SL45i do not support USSD and it should not according to the theory.

3.2.2 Servlets

To test if it was possible to get a connection between a servlet and a mobile phone MSc student Daniel Hellstrand made a HelloWorld.class and then made a link from a .wml file to the class. He placed the .class file in {tomcat home}/webapps/examples/WEB-INF/classes/ to be able to use the .class file he had to register it in {tomcat home}/webapps/examples/WEB-INF/web.xml. Had he placed it in another application (i.e. not in {tomcat home}/webapps/examples/), he would have had to make a new web.xml in WEB-INF directory, or added the code to a present web.xml file. The lines of code required in web.xml were:

```
<servlet>
        <description>Test servlet</description>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>HelloWorld</servlet-class>
</servlet>
```

After this he added a simple link from a .wml file which was directed directly to the .class file and when he connected to the homepage and tried the link he got the result “HelloWorld” from the HelloWorld.class file on a cell phone screen.

4 Analysis and recommendations

In this part I first analyse J2ME with different considerations. Then I analyse the business services I have been working with. From the analysis I have come to some recommendations. They are also presented in this part.

4.1 J2ME

Since the Java programs are placed in the mobile phones the mobile network will not be able to sense that it is direct communicating with a Java program, given that the program is not "hostile", containing viruses or provoking the system in other ways. Therefore, a Java program in a mobile device will not be a direct extra load itself to the network. However, Java programs are possible to stimulate a larger usage of mobile services in the long run, which means a higher load to the network. Therefore, I think it will affect the mobile network in the long run.

4.1.1 Distribution and billing

There will probably be a large number of Content Providers, providing Java programs for the mobile market soon. One problem for them is the marketing issue – How and where are they going to meet their customers? The problem for the users will be the corresponding issue – Where can they find the companies with the good Java programs? Another problem that might not be a big problem now, but may become an issue within the near future, is if people are starting to download Java applications containing viruses. I think that the customers want to be sure that the application being downloaded is not containing any viruses. The billing issue is also a problem for the content providers. How shall the customers pay them? In the figures below X could for instance be equal to 0.3.

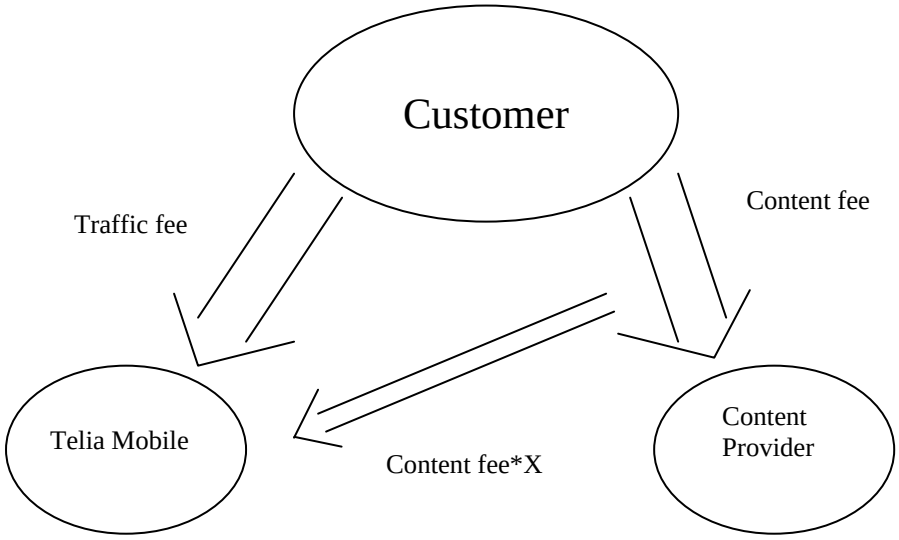

Figure 13. Fees paid by customer to mobile operator and content provider.

All of these problems can be solved by Telia Mobile by using a web portal, such as MyDOF or Halebop, from where the customer can find all Java programs on the market

divided into different categories: Games, Business, Maps, etc. From there the customer can download the applications and pay the content fee by their mobile telephone bill or pre-paid card. The customer would know that if one is looking for a Java program one always looks at the homepage of Halebop, for instance. Telia Mobile could check every program being offered at the homepage, so that no viruses are included. Then Telia Mobile could guarantee the customers that all programs being downloaded from Telia Mobile are not containing viruses. This would make the customer feel more secure and with more trust to the technology the usage is likely to be better off. Figure 13 above shows the principle of my proposal. One way to determine the value of X is to examine the Japanese market since it, specifically Docomo, has been using this kind of solution at their market for a long time.

To be more specific and to focus on the billing process I have focused on two solutions, which I think are the most appropriate. What have focused on an easy usage for the customer to stimulate the usage and to let Telia Mobile get control of the customers and the content providers.

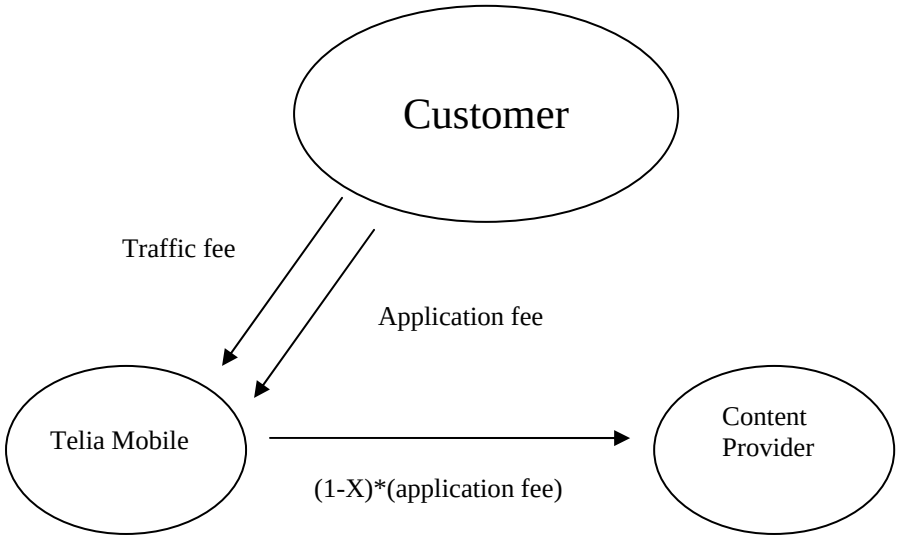

Figure 14. Proposal 1 of the billing process

The simplest technical solution is that the customer pays the traffic fee as usual and the fee for the program is being charged on the mobile phone bill. Afterwards Telia Mobile would pay (1-X)*(application price) to the content provider. The customer would only have to pay to one company, which is better. However, most customers are having problems estimating the traffic fee, i.e. the specific download cost. I think that this is one of the reasons why people are not using the GPRS technology much. Since the customer first has to pay for the application and also probably relates the traffic fee to the total application price, the customer does not know how much the application really costs. I am positive that this would lead to less usage. Clear information about the traffic costs with different GPRS subscriptions must be offered at the download site, so that the customers easily understand how much the application really costs. It is not enough to inform how much the cost is per byte. It must be easier to understand for the

customer. The cost must relate to something that the customer understands better, for instance is billing per time unit better in that sense since everybody can relate costs to the time. However, billing per time unit is not interesting in this case, but for each site where the application can be downloaded there should be calculations for different subscriptions and situations, e.g. how many bytes the customer has already downloaded during the month. This means that the customer can easily see how much the application approximately or exactly actually will cost.

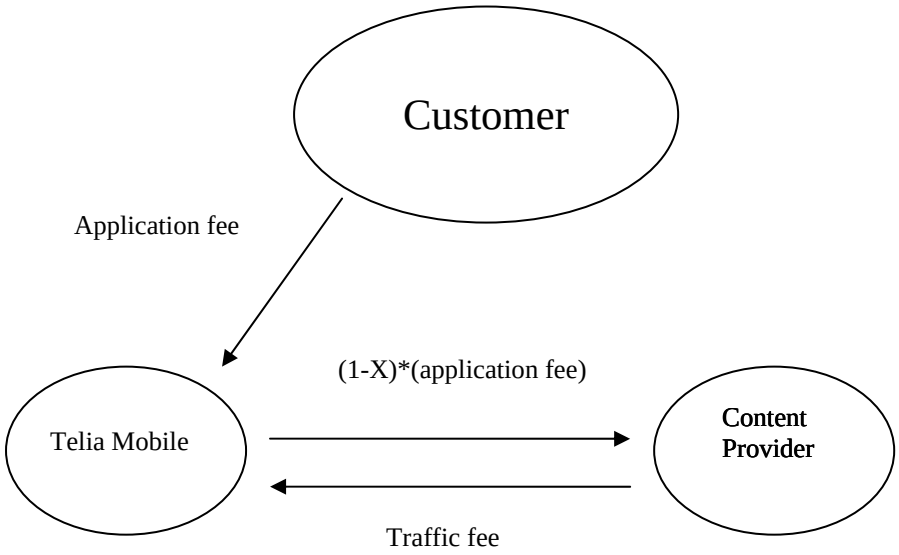

Figure 15. Proposal 2 of the billing process

The other alternative means that the customer only pays one bill to one company – Telia Mobile. The customer pays only one price for the application and no traffic fee is being charged. Instead, the Content Provider pays traffic fee to Telia Mobile and Telia Mobile pays (1-X)*(application price). Thus, the customer would pay the application price to Telia Mobile and Telia Mobile pays the Content Provider (1-X)*(application price) – (traffic fee). The good thing is that the customer only has to pay to one company and it is easy to understand the whole cost.

The best alternative is probably if the customer only pays one fee to one company – to Telia Mobile. After discussions with Mr. Hans Krantz at the Service Design Department at Telia Mobile, I found out that it is possible to solve the problem like this, but it takes a more complex technical solution than the first proposal and therefore it would mean more difficulties to implement it.

One problem that can occur is if the customer pays for an application and it turns out that he/she got the application for example for a Nokia instead of a Siemens and therefore, it does not work. Should the customer pay then? Telia Mobile mobile must discuss this issue and come to a conclusion. I think that a satisfied customer is better than an angry customer and therefore Telia Mobile in this case should offer the customers a new, correct application free of charge.

The above solutions miss an important issue: to attract content providers Telia Mobile has to offer them the biggest market as possible. This means that customers of other mobile operators must be incorporated in the solution. Thereby, the billing problem becomes even more complicated. The billing issue could be solved with help from Premium SMS or if the competitors on the market could agree on making a common solution/platform. It could be helpful to study the Norwegian Premium SMS market, since it is more developed than the Swedish one.

4.1.2 Lack of knowledge

Many new mobile phones being sold today are supporting Java and all of the coming new mobiles are most likely to support Java. However, among my friends and acquaintances, the only persons knowing what the technology can offer the customers and what it really means to them are other MSc. students. Therefore the mobile business must make an effort to change this. After several phone calls with people within the telecom business I have realized that the knowledge about Java technology in mobile phones is deficient. When not even a Technology Manager at a big international telecom company knows exactly what the technology is all about, how can the customers know?

One reason that there is a lack of knowledge concerning J2ME could be that the business is afraid of burning their fingers like they did with WAP. I think it is one of the reasons why Java has not been launched as a “technology of the future”. Another reason is that it takes time until all terminals are supporting Java. Java must offer something that increases the perceived value for the customer if this person shall buy a new mobile phone. Many new phones entering the market in Europe today are supporting Java, but it takes time until everybody has one. However, indications have been given by the mobile phone manufacturers that a majority of the phones being introduced to the market later this year (2003) will support Java. Maybe the business must wait till 3G has become popular, if it becomes popular, to see all customers having Java enabled mobile phones. If 3G becomes big business, I think many customers will buy a new mobile phone and most likely those mobile phones will support Java.

The lack of education concerning J2ME must be taken care of. Telia Mobile should send out information to the people working at all support departments to make sure everybody working with support knows and could explain to a customer what J2ME is all about. Another proposal is that Telia Mobile should have explanations and links from their homepage and from a web portal. The basic idea of Java should be explained – that the customer can download programs to the mobile phone like a program can be downloaded to a PC. The customers must be informed what the Java technology can offer them, how it can increase the perceived customer value. It shall also be presented how it is being done and how much it costs. There should be separate links for developers and customers. On the developer site there should be links to Sun from where one can download the WTK, more advanced descriptions of the technology, and links to different mobile phone manufacturers from where one can download the additional APIs. I wrote Sun an email and proposed that they should do the latter, since it would be good as a developer to have all information needed on one homepage. However, they answered that it was not in their interest to have such a site in their domain. Telia Mobile is one of the actors that can make money on Java. Therefore, Telia Mobile should put an effort to stimulate the developers to produce good Java

applications. One way could be to let Telia Mobile be the company that offers the natural site to search information for developers. Another way could be to arrange competitions among young people, preferably university students, where the best Java applications would be awarded. This would not only create interesting applications, but also spread knowledge of J2ME and make people aware of the technology.

4.1.3 Japan

In Japan the Average Revenue Per User, ARPU, increased very rapidly when the Java technology in mobile phones was first introduced. After a while the ARPU for customers with Java enabled phones sank down a bit, but the level was still significantly higher than for a normal customer and it still is (twice as high). However, in Japan the customer is happy when a cartoon figure is jumping up and down, singing, but in Sweden the customers are likely not willing to pay anything for such services. The services must probably be more sophisticated.

4.1.4 Development of J2ME and JSR

While I have been writing this report new information, such as new APIs, reports, about J2ME has been published continuously on the Internet. Many things are now possible that were not possible six months ago. Since the development is very fast Telia Mobile should have someone or somebody who could control the new information about J2ME. It would for instance be very good to know for service designers if another service packet would be released by Sun that would support USSD. Therefore, I think that someone or somebody within the company should be responsible to keep himself/herself concerned with the latest news within the technology. This person should spread new important information to interested people. Besides, the developers would know who to turn to if/when they get questions about J2ME.

Telia Mobile should be an active part in the JSR and thereby looking out for their own interests. They should have ongoing investigations that look into what new features are needed to satisfy the needs of the customers and the developers.

4.1.5 Games

Since the computer gaming industry is very young and from the beginning focused on young people it’s still considered childish to play these games. With time, however, I personally think it will be more accepted to play these games among adults. The kids that are playing them today are likely to play when they grow older and new generations will grow up with these games as a “natural ingredient” of life. Therefore, possible cash cows for the future are the network games. The development of the computer game industry tends to develop towards network games where the users play against each other. It is very likely that this development will look the same for the mobile games. With better screens, better bandwidth and better games (Java games that can be downloaded over the Internet instead of being included in the phone when it is being bought) there is a big opportunity for the mobile operators to make money in form of a big increase of data traffic being sent over the mobile network. In fact, you can already today play mobile network games, such as Othello.

4.1.6 What APIs are included?

When services are developed one has to consider which functions are going to be used. Are there only standard features, which are supported in MIDP, or are there special functions, which demand the presence of optional packages. The use of optional

packages will definitely limit the number of devices able to use the application, which obviously is not good, but in some cases one can make one piece of software for the optional package devices and one piece of software for the devices without the optional software.

4.1.7 Future services

For future services like mobile reference the carrier should be SMS (or USSD when it is supported by MIDP) instead of calling. The usage can be made easier if there is support for Java interfaces, SMS, but not USSD for the moment. It goes faster to activate features/services with SMS within Java applications, than to use calling as bearer. It is also important that Telia Mobile have in mind that MIDP only supports numbers and not any other symbols. Therefore, when one is designing new services it's better if features such as mobile reference can be activated only by numbers and for instance not by # or *.

From the results from the Dirigent and Centrex service, I have discovered many limitations, such as the phones must have the right API for the service in question, the service must be controlled with calling as bearer and the codes should not contain anything but numbers. To solve the problem one could change the interface through which the services are controlled, and instead of controlling them via calling, SMS or USSD, HTTP could be used. All Java enabled phones could then communicate with the service since HTTP connection is supported in MIDP.

Today, there are no size limitations of the applications being downloaded over the GPRS network. However, there is a limitation of 60 kbytes in the WAP Gateways for uploading. The sizes of games for Samsung phones are about 50 kbytes and the application for Dirigent consists of only 4 kbyte data. Therefore, today it is not a big problem to upload Java applications to other mobile devices considering the sizes of the applications. On the other hand it could be a problem for many customers if they can't upload more than 60 kbytes. Telia Mobile should consider this when designing a strategy to prevent customers from copying Java applications between one another.

4.1.8 USSD and development of new mobile services

USSD is not supported and it's being used as carrier for some services, i.e. mobile extension. If USSD was supported it would be easy to make a Java application that would set a mobile extension depending on the time and so on. For the customer it would be easier than typing a long code that is very easy to forget. It would also be good for the service developers at Telia Mobile since they could use this in future services.

4.1.9 Servlets and JSPs

Servlets and JSPs enable mobile services that today are only offered for PC users. Many interesting applications can be developed that would make the world more mobile.

4.1.10 Java Card Technology

Java Card Technology enables applications to be placed on the SIM card and that creates many possibilities. It also includes many good cryptographic alternatives. The possibility to use PKI enables the customers to use the mobile phone to do banking services, such as paying bills and checking the balance. Another example is that the

customers will be able to read their e-mail accounts from the job securely. These things will also be possible to do with J2ME, but the question occurs where the private key shall be placed – in the phone or on the SIM card, i.e. who will have the control of the customer – the mobile phone manufacturer or the mobile operator? This will probably become an issue within the near future. However, there is no API today that supports communication between a Java Card and a J2ME application. I suppose it is just a matter of time till there will be one. Then synergies with the two technologies can be gained. For instance, a private key could be placed on the Java Card, which would give the mobile operator the control of the customer. The customer could then with help from Java applications placed in the phone read and send restricted emails from work securely or make secure bank transaction.

4.1.11 Different applications for different mobile phones

Unfortunately, it is a fact that the mobile device producers let the J2ME-application developer use more or less of the functions in the mobile phone in terms of API. With Siemens' APIs one gets the APIs concerning Calling, SMS and Phonebook, which enable the application developer to create programs that can use the mentioned services. With Nokia one can only use the SMS feature, of the features above, with the help of Nokias APIs. That means that possible services and features that could have been created to Nokia mobile phones are not possible to develop e.g. all services using calling as a carrier. Furthermore, the developers have to develop different applications for mobile phones from different manufacturers. This conflicts with one of the genius idea about Java – "It should be platform independent". Hopefully, this will change with time when new versions of MIDP will be developed.

4.1.12 WMA

The WMA APIs can be downloaded today. However, there is no mobile phone on the market that is supporting WMA. Nokia 3650 is likely to be the first one. It is supposed to be out on the market in February 2003. (http://www.mobil.se/nyheter/visa.asp?id=6230&sid=1, 2003-02-06). Therefore, it will take some time before many customers are having Java enabled phones that support WMA.

4.1.13 MIDP 2.0 and MP3

MIDP 2.0 demands 100 kbytes more memory than MIDP 1.0 does. Therefore, MIDP 2.0 needs more memory of the phone. The Media API supported by MIDP 2.0 enables the possibility to create MP3 players and other players supporting other sound formats. Since the downloading of MP3 songs on the Internet is extensive it is likely that the customers will start downloading MP3 songs over the mobile network. This might generate large profits to the mobile operators if the users start to download MP3 songs in a big extent.

4.1.14 Storage space online

Today there are virtual hard drives, which you can utilize if you have a regular computer. This virtual hard drive can be used to store information on, such as movie clips, music, pictures, documents and so on. It can be accessed from any computer around the world and it makes it possible to travel light, without carrying any diskettes, laptops or CDs. This would be an excellent service for mobile phones since storage is a crucial element due to its high price range. The service would enable people to access their storage area both from a regular computer with an Internet connection and from a

mobile device through a GPRS or WAP session. Appealing and easy to use interfaces could be made in Java to enable the user to navigate the storage area just as easy on the mobile device as he would on a PC. On the storage area one could keep a picture gallery, documents, link collection, music and movies that could be streamed to the mobile device if the user wished to utilize them.

This storage area is intended for a single user and each user should have his/her own area. But as a complement one could have a storage area, which is used by a group, for example a family, where a web board could be implemented. This web board could be used by the family to write messages such as, buy milk, feed the cat, walk the dog or "I will be home by six" etc.

4.2 Mobile business services

4.2.1 Dirigent

The problem with the mobile reference feature is the lack of simplicity. Not even the people at the service design department at Telia Mobile know the codes that one uses to activate a mobile reference. With the application I have developed the setting of a mobile reference gets a lot easier.

4.2.2 Centrex

To change the string that sets a mobile reference with a Centrex subscription *, # and p (pause symbol) to just numbers would take time and would cost for Telia Mobile. Therefore, it will not be possible to set a mobile reference with a Java application as long as the calling API is not supporting *, # and p (today there is only one – Siemens').

4.3 Security

4.3.1 Trojan horse

One scenario could be a "trojan horse program", which could call, for instance in the middle of the night, an unauthorized Premium Service, e.g. 071-XXXXXX in Sweden, so that the subscriber would be charged for something he/she did not take advantage of. However, using the Siemens calling API one always has to confirm when calling. This would prevent the virus from executing unless an exploit is found in the API. Today, this is the only API for calling on the market. However, if a calling API without the demand of confirming a call with the mobile phone would enter the market, Trojan horses could start to appear. For the analogue reason the API for Siemens' phonebook is restricted (one can only use already called numbers). If a mobile virus could get into the mobile phone it could change the number of for instance "home" or "mum" or why not all names in the address book to Premium Service numbers.

4.3.2 Data erased

It could also be a program that would erase data for the user. But as of today the MIDlets executed are run in a safe environment that are not supposed to get a hold of sensitive material unless manufacturers decide to make this available through optional APIs.

4.3.3 Copying of applications

In todays fast moving market where software is used to accomplish every task it is of vital importance that this software is protected from being copied. Since the pirate copying industry is widely spread on the pc market, using internet as a base to stand on, it is likely that the same thing will happen with mobile phones and the programs developed for them. This is why it is of great importance that manufacturers unite to develop some kind of copying protection for the future applications in mobile devices.

Applications can already be copied on cellular phones and PDAs. The one I and MSc student Daniel Hellstrand have confirmed is Siemens SL45i, which makes it possible to transfer Java applications via cable connected to a PC. It is also feasible that applications can be transferred using IR, MMS, bluetooth, and e-mail within a short period of time (if it is not already possible). Here are some thoughts on how it can be stopped.

- I think that one way to prevent copying of Java applications would be to make applications that bind to hardware specific attributes in the device such as component serial numbers, IMEI (terminal specific) or ICC (SIM card specific) but after further research I have come to the conclusion that this will probably not be possible today since these codes cannot be accessed through a standard MIDP API. However, it is feasible that manufacturers choose to include this as an optional API in their devices, which would enable the opportunity of easy authentication.

- When making the application, authentication codes could be integrated with the source code and compiled, these codes could later be used for authentication of users. This has its drawbacks since the application could be decompiled and the codes revealed, unless they were encrypted.

- Put a password in the .jad file (contains information about the Java program) as a vendor specific property, this password should be a hashed value which is retrieved with MIDlet.getAppProperty() (a function that is being used to able to see the information in the .jad file), and the password should be distributed to the user at a different time than the download occasion. A small drawback is that you have to produce a new .jad file for every download but this is better than including the password in the source code and compiling it since that is more time consuming and could be retrieved by decompiling.

- When a program is downloaded, an authentication via http could be used on a regular basis to check the validity of the program.

- The manufacturers could disable the ability to transfer programs via infra, bluetooth, MMS or e-mail. However, people would soon find ways around this by renaming or packaging the applications differently, so this is not a good solution.

- Only authenticate the user when the download is done OTA (Over The Air). This standpoint will not stop a customer from spreading an application. It assumes the general attitude toward piracy is that happy customers will compensate the loss of revenues. The customer doesn’t have to deal with

malfunctioning applications due to complicated authentication methods and download procedures.

- SonyEricsson P800 uses a cryptographic authentication method for verifying the .sis (Symbian Standard Installation) file during the installation. The verification procedure identifies a digital signature and a certificate in the .sis file and compares this to a root certificate in the phone. A successful verification tells you that you can trust the content of the .sis file and that it has not been altered since the manufacturer compiled it. (http://www.sonyericsson.com/downloads/P800_UG_R1a_SV.pdf, 2003-01-22) This might be one way to solve the problem.

4.3.4 WAP Gateways

Today, there are no size limitations of the applications being downloaded over the GPRS network. However, there is a limitation of 60 kbytes in the WAP Gateways for uploading. The sizes of games for Samsung phones are about 50 kbytes and the application for Dirigent consists of only 4kbyte data. Therefore, today it is not a big problem to upload Java applications to other mobile devices considering the sizes of the applications. On the other hand it could be a problem for many customers if they can't upload more than 60 kbytes. Telia Mobile should consider this when designing a strategy to prevent customers from copying Java applications between one another. Virus scanning in gateways can be bought when/if people start to download Java applications in a big extent.

5 Conclusions and recommendations for further studies

In this last part I present my conclusions and recommendations for further studies.

5.1 Conclusions

The Java technology for mobile phones, J2ME, is still young. The basic idea of Java is that it is a technology independent from the platform where it is being run. However, todays APIs included in the application environment, MIDP, are not enough. Since many applications just work for a specific mobile manufacturer J2ME interferes with the basic idea of Java. There are many enabled Java phones on the market today, but most customers have a phone that is not supporting Java. I do not think people in general buy new phones to get one that enables Java, but in a few years I think that most people have bought a new 3G phone and the phones are likely to support Java. I think the Java market will extend significantly with the introduction of 3G. Better bandwidth and new phones will enable new, better and more sophisticated services.

I think that the ARPU in Europe will also be bigger for customers with Java enabled phones as in Japan and South Korea, even though the cultural differences are big. I think the step from downloading programs to PCs to downloading programs to mobile devices is natural and I think the Java technology is the appropriate technology to do it with.

In the short term the technology suffers from being too young. Desired APIs are not available or supported and therefore J2ME is maybe not aimed to business services.

I think the biggest problem with the technology in the long run is that many copies of Java programs will fluctuate on the market. Since pirate programs for PCs are so common, why will we not see the same development within the mobile business?

J2ME has come to stay. Today it is natural to download programs to a PC and tomorrow we will do the same thing with our mobile phones. However, tomorrow is already here.

5.2 Recommendations for further studies

Some of the things I have not been able to examine because of my delimitations could be examined as another master's thesis project. Therefore, I list two purposals for further studies.

- Investigate the difference between J2ME and other technologies such as SIMAT (Subscriber Identity Module Application Toolkit), Symbian and other competing technologies. Are they really competing or are they just supplements to one another?

- Examine how the Java technology in mobile devices affects mobile operators considering games in general and network games in particular.

6 References

6.1 Printed book

JOHN W. MUCHOW; C*ore J2ME technology & MIDP*; Prentice Hall 2002; ISBN: 0-13-066911-3
Introduction to the J2ME technology and MIDP

6.2 Internet

DAVID BANK; *Java saga*:1995;
http://www.wired.com/wired/archive/3.12/java.saga.html, 2002-09-25
Java history

STEPHANIE BODOFF; *JavaServer Pages Technology*; 2002;
http://java.sun.com/webservices/docs/1.0/tutorial/doc/JSPIntro.html, 2002-11-05
Introduction to JavaServer Page (JSP)

JON BYUOS: *Java technology: an early history*; 1998;
http://java.sun.com/features/1998/05/birthday.html, 2002-06-07
Java history

ERIC GIGUERE; *J2ME Optional Packages;* 2002;
http://wireless.java.sun.com/midp/articles/optional/ 2002-12-18
Article about optional packages, which deals with the different aspects of it. Why are there optional packages, how are they used etc.

ANNA HEDBERG; *Nokia 3650 släpps inom kort*; 2003
http://www.mobil.se/nyheter/visa.asp?id=6230&sid=1, 2003-02-06
Information about the release of Nokia 3650

DAVID HEMPHILL; *J2ME Gets Personal;* 2002;
http://www.fawcette.com/javapro/2002_12/magazine/features/dhemphill/default_pf.asp
Image and information about J2ME, CDC, CDLC

JONATHAN KNUDSEN; *What's New in MIDP 2.0?*; 2002;
http://wireless.java.sun.com/midp/articles/midp20/, 2003-01-08
News in MIDP 2.0 compared to MIDP 1.0

C.ENRIQUE ORTIZ;*A Survey of J2ME Today;* 2002;
http://wireless.java.sun.com/getstart/articles/survey/, 2003-01-30
A survey of J2ME today. A consolidated, big-picture view of where the J2ME platform stands today. It explains the terminology you need to know to make sense out of an increasingly complex topic.

MICHAEL JUNTAO YUAN & JU LONG; *Security challenges and solutions for mobile commerce applications*; 2002;
http://www-106.ibm.com/developerworks/webservices/library/wi-secj2me.html?dwzone=webservices, 2003-01-29
Security challenges and solutions for mobile commerce applications

http://java.sun.com/java2/whatis/, 2002-09-26
Short explanations about the Java language in general and some about the different platforms J2SE, J2EE, J2ME

http://java.sun.com/j2me/j2me-ds-0201.pdf, 2002-09-26
Short introduction to J2ME, which covers CDC, CLDC, MIDP and Personal Profile

http://java.sun.com/products/cldc/, 2002-09-26
Short introduction to CLDC, which also addresses the two available virtual machines K Virtual Machine and CLDC Hotspot Virtual Machine

http://java.sun.com/products/servlet/, 2002-11-05
An introduction to the Java servlet technology

http://java.sun.com/products/javacard/datasheet.html/, 2002-11-13
Introduction to Java card technology

http://wireless.java.sun.com/getstart/articles/survey/survey_fig1.gif, 2003-01-30
Image

http://e-www.motorola.com/webapp/sps/site/application.jsp?nodeId=03M0ym4sDZxM0zbxntsZ, 2003-01-30
Image

http://www.nokia.com/java/for_developers.html, 2002-10-15
Introduction to J2ME

http://www.nokia.se/mobiltelefoner/7650/#, 2003-02-03
Image

http://www.my-siemens.com/MySiemens/CDA/Index/0,2730,SE_sv_0_product%253AMW%252FHD%252FHD%252FSL45I%252Fdesc,FF.html, 2003-02-03
Image

http://www.sonyericsson.com/downloads/P800_UG_R1a_SV.pdf, 2003-01-22
User manual for the mobile device SonyEricsson P800. I used this manual to search for information on how to transfer files to and from a P800, and I also tried to find out which Java packages were included

http://www.sumea.com/6_0/games.html, 2003-01-30
Image

http://www.etsi.org/, 2003-01-08
Telecom standards

http://www.jcp.org/en/jsr/stage, 2002-11-20
Description of the Java Specification Request process

http://www.jcp.org/en/jsr/detail?id=82, 2003-01-20
JSR 82, Bluetooth

http://www.jcp.org/en/press/success/bluetooth, 2003-01-20
Discussion about the development and the timeline of JSR82, Java API for Bluetooth

http://www.ipa.go.jp/security/fy13/report/mobile_security/java-keitai.pdf, 2002-10-25
Experiences of mobile viruses in Japan

http://www.nyteknik.se/pub/ipsart.asp?art_id=20050, 2002-12-16
Article about Java in mobile phones

6.3 Personal communication

MSc. Erik Björk, Service Design, Telia Mobile AB
MSc. Lars Dovner, Service Design, Telia Mobile AB
MSc. Ola Edström, Service Design, Telia Mobile AB
Doctor Henrik Eriksson, NADA, KTH
MSc. Lars Eriksson, Service Design, Telia Mobile AB
MSc. student Daniel Hellstrand, Luleå University of Technology
MSc. Gunnar Hägglund, Service Design, Telia Mobile AB
Mr. Hans Krantz, Service Design, Telia Mobile AB
MSc. Lars Liljestam, Service Design, Telia Mobile AB
MSc. Markus Marklund, Service Design, Telia Mobile AB
MSc. Peter Sirviö, Service Design, Telia Mobile AB
MSc. Mikael Söderberg, Service Design, Telia Mobile AB
MSc. Robert Tjernström, Service Design, Telia Mobile AB
MSc. Tommy Ytterström, Service Design, Telia Mobile AB

6.4 Other

Embassy of Sweden, Science & Technology Office, Sweden-Japan IT Program,
Internet security issues, Sabine Ehlers, 2002-07-25
GPRS Protocols and Procedures, Enea Epact
GSM System Overview, APIS Training & Seminars
MIDP Style Guide, Mobile Information Device Profile (MIDP), 1.0a, August 2002
Tjänstebeskrivning ”Dirigent”, Telia Mobile AB
Tjänstebeskrivning Mobil anslutning Direkt i Telia Centrex

7 Appendix

Common Packages	J2SE 1.3.1	CDC 1.0	CLDC 1.0	CLDC 1.1	MIDP 1.0	MIDP 2.0
java.applet	X					
java.awt	X					
java.awt.color	X					
java.awt.datatransfer	X					
java.awt.dnd	X					
java.awt.event	X					
java.awt.font	X					
java.awt.geom	X					
java.awt.im	X					
java.awt.im.spi	X					
java.awt.image	X					
java.awt.image.renderable	X					
java.awt.print	X					
java.beans	X					
java.beans.beancontext	X					
java.io	X	X	X	X	X	X
java.lang	X	X	X	X	X	X
java.lang.ref	X	X		X		
java.lang.reflect	X	X				
java.math	X	X				
java.net	X	X				
java.rmi	X					
java.rmi.activation	X					
java.rmi.dgc	X					
java.rmi.registry	X					
java.rmi.server	X					
java.security	X	X				
java.security.acl	X					
java.security.cert	X	X				
java.security.interfaces	X					
java.security.spec	X					
java.sql	X					
java.text	X	X				
java.util	X	X	X	X	X	X
java.util.jar	X	X				
java.util.zip	X	X				
javax.accessibility	X					
javax.microedtion.io		X	X	X	X	X
javax.microedition.lcdui					X	X
javax.microedition.lcdui.game						X

javax.microedition.media						X
javax.microedition.media.control						X
javax.microedition.midlet					X	X
javax.microedition.pim						
javax.microedition.pki						X
javax.microedition.rms					X	X
javax.microedition.xlet						
javax.microedition.xlet.ixc						
javax.naming	X					
javax.naming.directory	X					
javax.naming.event	X					
javax.naming.ldap	X					
javax.naming.spi	X					
javax.rmi	X					
javax.rmi.CORBA	X					
javax.sound.midi	X					
javax.sound.midi.spi	X					
javax.sound.sampled	X					
javax.sound.sampled.spi	X					
javax.swing	X					
javax.swing.border	X					
javax.swing.colorchooser	X					
javax.swing.event	X					
javax.swing.filechooser	X					
javax.swing.plaf	X					
javax.swing.plaf.basic	X					
javax.swing.plaf.metal	X					
javax.swing.plaf.multi	X					
javax.swing.table	X					
javax.swing.text	X					
javax.swing.text.html	X					
javax.swing.text.html.parser	X					
javax.swing.text.rtf	X					
javax.swing.tree	X					
javax.swing.undo	X					
javax.transaction	X					

Table 5. Shows the difference and similarities between the various configurations and profiles. Modified version of the original.
(http://wireless.java.sun.com/getstart/articles/survey/, 2002-12-20)